Informe Detallado: Implementación de Autenticación y Autorización en una Aplicación Web

1. Objetivo del Ejercicio

El objetivo principal de este ejercicio fue implementar un sistema básico de autenticación y autorización en una aplicación web, permitiendo a los usuarios registrarse, iniciar sesión y acceder a funcionalidades específicas según su rol. El escenario propuesto fue un sistema de gestión de contenido para un blog con dos tipos de usuarios: Administradores (con acceso total a publicaciones) y Usuarios comunes (con acceso solo de lectura).

2. Herramientas Utilizadas

Para esta implementación, se utilizaron las siguientes herramientas y tecnologías recomendadas:

- **Backend:** Node.js con el framework Express.js
- **Hashing de Contraseñas:** bcrypt para el almacenamiento seguro de contraseñas.
- **Base de Datos:** MongoDB Atlas para la persistencia de datos de usuarios y roles.
- **Gestión de Sesiones:** express-session para manejar las sesiones de usuario.
- **Gestión de Base de Datos (ORM/ODM):** Mongoose para la interacción con MongoDB.

3. Implementación

Autenticación Básica con Contraseñas

Se implementó un flujo completo de registro e inicio de sesión seguro:

- **Página de Registro:** Se creó una interfaz HTML (register.html) donde los usuarios pueden ingresar su nombre de usuario y contraseña.
- **Hashing de Contraseñas:** Al registrar un nuevo usuario, la contraseña proporcionada se hashea utilizando el algoritmo bcrypt antes de ser almacenada en la base de datos. Esto asegura que las contraseñas nunca se guarden en texto plano, aumentando significativamente la seguridad. Se generó un "salt" (bcrypt.genSalt(10)) y luego se hasheó la contraseña (bcrypt.hash(password, salt)).

- **Registro en MongoDB:** El nuevo usuario, con su contraseña hasheada y un rol por defecto de 'usuario', se guarda en una colección users en MongoDB Atlas.
- **Página de Inicio de Sesión:** Se desarrolló una página (login.html) donde los usuarios introducen sus credenciales.
- **Verificación de Credenciales:** Al iniciar sesión, el servidor busca el usuario por nombre de usuario en la base de datos y compara la contraseña proporcionada con la versión hasheada almacenada utilizando bcrypt.compare(). Si las credenciales son válidas, se crea una sesión para el usuario y se le redirige al dashboard.

Creación de Roles de Usuario

Se estableció un sistema de roles básico:

- **Rol por Defecto:** Cada nuevo usuario registrado se le asigna automáticamente el rol de 'usuario'.
- **Asignación de Roles (Administrador):** Para asignar el rol de 'administrador', se realizó un cambio manual directamente en la base de datos de MongoDB Atlas. Esto demuestra la capacidad de tener roles diferenciados y su persistencia.
- **Modelo de Usuario:** Se definió un esquema userSchema con mongoose, incluyendo campos para username, password (hasheado) y role.

Restricción de Acceso según el Rol

El acceso a diferentes partes de la aplicación se gestiona en función del rol del usuario autenticado:

- **Middleware isAdmin:** Se creó un middleware isAdmin que verifica el rol del usuario almacenado en la sesión (req.session.user.role). Si el rol es 'administrador', permite el acceso (next()); de lo contrario, deniega el acceso con un estado 403 (Prohibido).
- **Dashboard de Usuario (/dashboard):** Un usuario común puede acceder a su panel, el cual muestra su nombre de usuario y su rol. Desde aquí, puede intentar acceder al panel de administrador, pero será redirigido si no tiene el rol correcto.
- **Panel de Administrador (/admin-panel):** Esta ruta está protegida por el middleware isAdmin. Solo los usuarios con el rol de 'administrador' pueden acceder a este panel, donde se les presenta un menú con opciones para la gestión de publicaciones.

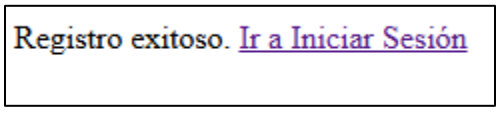
- **Funcionalidades de Gestión de Publicaciones:** Rutas como /publicacion/crear (POST), /publicacion/editar/:id (PUT) y /publicacion/eliminar/:id (DELETE) también están protegidas con el middleware isAdmin. Esto asegura que solo los administradores puedan "crear, editar y eliminar publicaciones" (simuladas en este ejercicio). Al hacer clic en los botones correspondientes en el panel de administrador, se envía una solicitud a estas rutas, y el servidor responde con un mensaje simulado.

4. Pruebas y Verificación

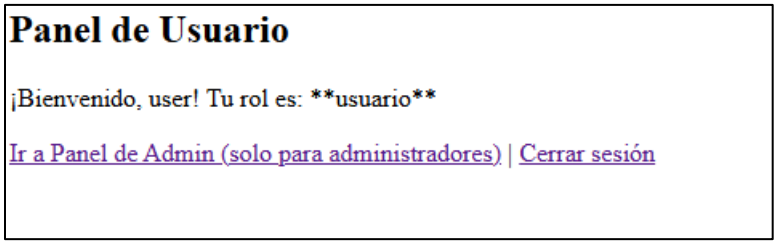
Se realizaron las siguientes pruebas para verificar la correcta implementación:

- **Flujo de Registro y Login:**
    - Un usuario fue registrado exitosamente, con la contraseña hasheada y almacenada en MongoDB.

```
Servidor escuchando en http://localhost:3000
Ve a http://localhost:3000/register para registrarte.
Usuario registrado: {
  username: 'user',
  hashedPassword: '$2b$10$NDyYnD47kL01GLxkqB1zSeUfn8ShZcIN8PUlj2s75/dysCE0feXDS'
}
```



    - El usuario pudo iniciar sesión y fue redirigido al dashboard de usuario, confirmando su rol de 'usuario común'.

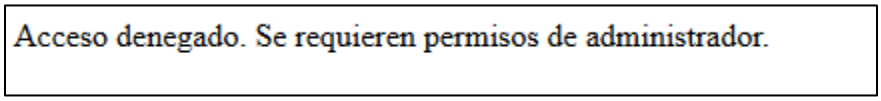


- **Acceso según Roles:**
    - El rol del usuario fue modificado manualmente a 'administrador' en MongoDB Atlas.

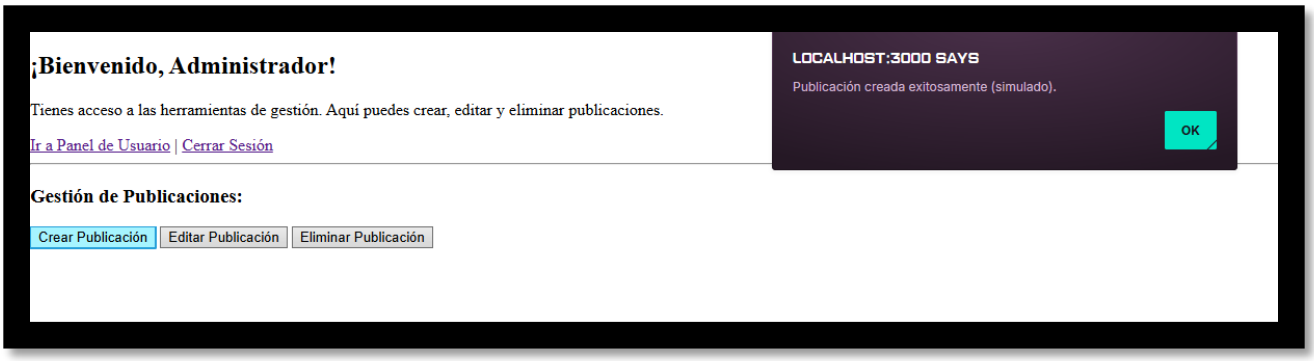
```
_id: ObjectId('685e2806cb42f491476a8bcd')
username : "admin"
password : "$2b$10$upb8Q/roJ15yGnilqV7rIOQk0c/4SkYA7awmh3vTgHiCjGfWkp/GK"
role : "administrador"
__v : 0

_id: ObjectId('68633ede8e2f1cdae513d967')
username : "user"
password : "$2b$10$Jgnn3CB3/CIMBfNDWttFYODNHyKtwc1KiPE0F5AhxN3xPqB/c3IfG"
role : "usuario"
__v : 0
```


- Al iniciar sesión con el usuario ahora como administrador, se logró acceder al /admin-panel, que muestra los botones de gestión de publicaciones.

- Al intentar acceder a /admin-panel como usuario común (sin modificar el rol), se recibió un mensaje de acceso denegado, verificando la restricción.



- Se hizo clic en los botones de "Crear Publicación", "Editar Publicación" y "Eliminar Publicación" en el panel de administrador, lo que generó mensajes de confirmación simulados, demostrando que las rutas protegidas para la gestión de publicaciones están accesibles solo para el administrador.



5. Conclusión

Este ejercicio permitió implementar con éxito un sistema robusto de autenticación y autorización utilizando Node.js, Express.js y MongoDB. Se logró un registro seguro de usuarios con hashing de contraseñas, un sistema de roles (usuario y administrador), y una efectiva restricción de acceso a las funcionalidades según el rol del usuario, cumpliendo con todos los requisitos del ejercicio práctico. La persistencia de datos a través de MongoDB Atlas asegura que los usuarios y sus roles se mantengan a lo largo del tiempo.